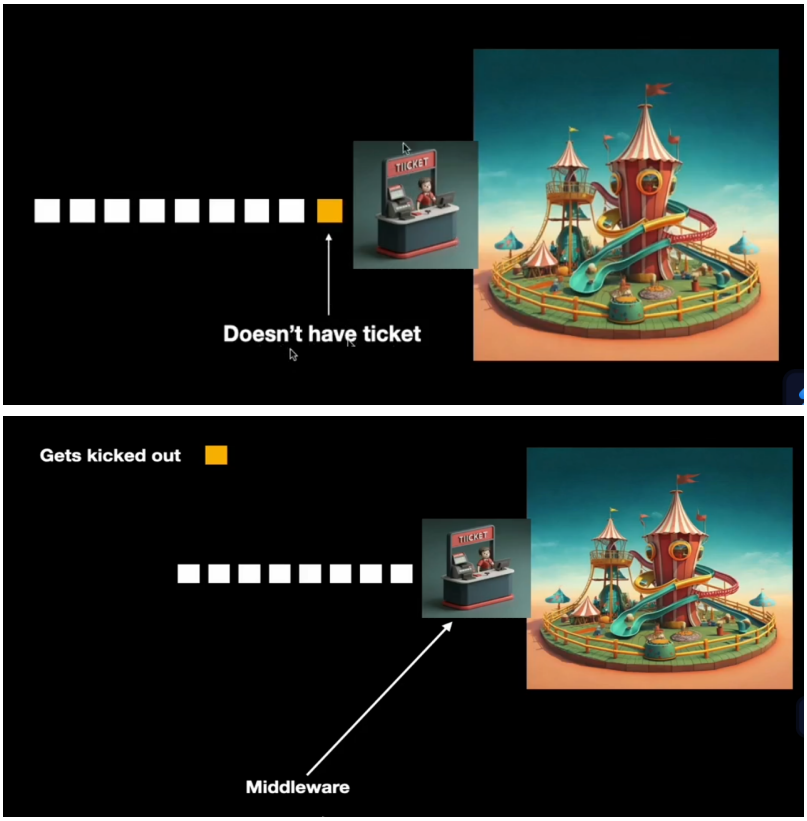
Middleware with real world example

Take the case of **amusement part**



The ticket checker is the **middleware -> A person sitting between your end client and your final ride that you are trying to access doing certain checks making sure only people that have legitmate things done can only pass through**

What is Middleware ??

-> `Express` is a routing and middleware web framework of its own.

🔷 **An `express` is essentially a series of middleware function calls**

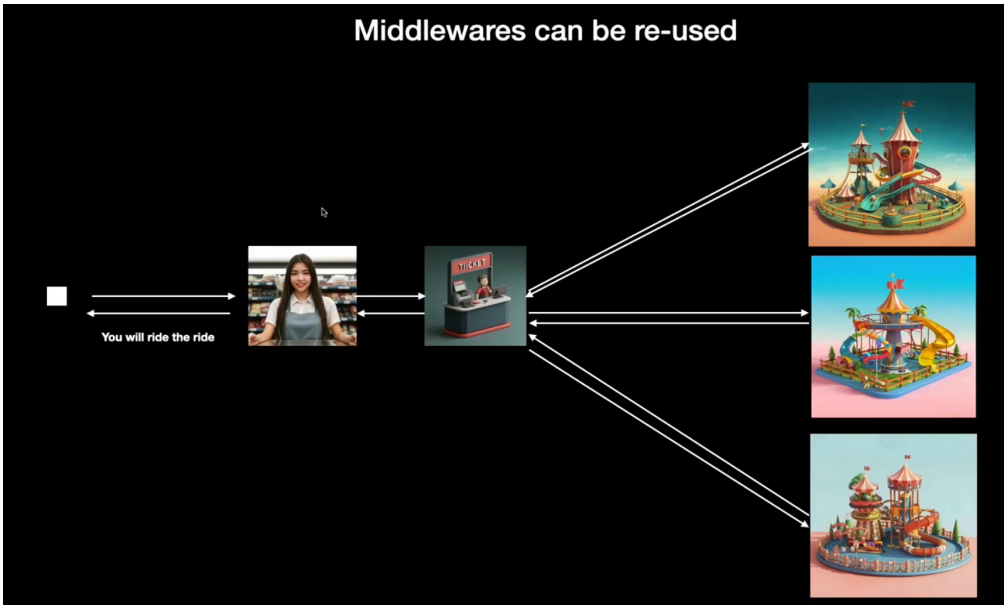
`Middleware` are functions that have access to **request object (req),response object (res)** and the **Next** `Middleware` function in the application's `request - response` cycle

`Middleware` functions can perform the following tasks ->

- Execute any code
- Make changes to the **request** and **response** objects
- End the **request - response** cycle
- call the next `middleware` function in the stack

An `Express` app is essentially a series of `middleware` calls

💡 Why Middlewares ??



A single ticket checker can handle multiple amusement park instead of each ticket checker kept for single amusement park

Lets codify this example

```
const express = require("express")

const app = express();

app.get("/ride1", function(req, res){
    res.json({
        msg : "You have successfully ridden ride1",
    })
})

app.listen(3000);
```

But the above code has one problem we have never checked whether the person has **Ticket or not ?? or certain age or not ??**

lets introduce them

```
// function that will return a boolean if the age of the person is more than 14
```

```
function age14OrMore(age){
    if(age < 14){
        return false;
    }else{
        return true;
    }
}
```

Now implying it in the main code

```
const express = require("express")

const app = express();


function age14OrMore(age){
    if(age >= 14){
        return true;
    }else{
        return false;
    }
}

app.get("/ride1", function(req, res){
    if(age14OrMore(req.query.age)){
        res.json({
            msg : "You can ride the ride1"
        })
    }else{
        res.status(411).json({
            msg : "You are not eligible due to the age"
        })
    }
})

// similarly you can use the same ticket checker for ride2 as well

app.get("/ride2", function(req, res){
    if(age14OrMore(req.query.age)){
        res.json({
            msg : "You can ride the ride2"
        })
    }else{
        res.status(411).json({
            msg : "You are not eligible due to the age"
        })
    }
})


app.listen(3000);
```

The above is the **one** way to do that but the **Better way is to do it via** `Middleware`

see the syntax carefully

How to define the `Middleware` ??

```
function age14OrMoreMiddleware(req, res, next){
    if(age >= 14){
        next();
    }else{
        res.json({
            msg:"Not eligible"
        })
    }
}
```

In the above code we know about the `req`, `res` but **What the hack is** `next`**??**

-> go to the **Definition of** `Middleware` **written above**

-> `Middleware` are functions that have access to **response object, request object** and the **Next** `Middleware` function in the application's `request - response` cycle

> ⚠ Every `Middleware` or functions you write in `express` has actually **3 arguments (req, res, next)**
>
> > we could have passed third argument while defining routes `/ride1` function (**next**) **but as it was not needed so**
> >
> > > Mostly you need only 2 arguments (`req, and res`) so you could have passed `next` in all the route handlers written above

> ⚠ `next()` **helps to propagate to the next middleware**
>
> > ⚠ **Remember to not pass anything inside next() as argument as** `Express` **automatically resolves this**

The Biggest advantage of using `Middleware` **is to make them what they really are supposed to do**

For example -> the current code looks like this BUT...

```
app.get("/ride1", function(req, res){
    if(age14OrMore(req.query.age)){
        res.json({
            msg : "You can ride the ride1"
        })
    }else{
        res.status(411).json({
            msg : "You are not eligible due to the age"
```

```
        })
    }
})
```

:sparkle: **Very Very important point**

The role of `/ride1` was to **give the ride whoever comes its duty is NOT to check that whether its age is eligible or not just make him / her ride ?? thats where the role of `middleware` comes**

It (ride1) should look somthing like this

```
app.get("/ride1", function(req, res){
    res.json({
        msg : "successfully done ride1",
    })
})  // check ka kaam kyu kre ye ??
```

as we have already defined the `middleware` above so lets see how to use it ??

How to Use the Middleware ??

```
const express = require("express")

const app = express();

function age14OrMoreMiddleware(req, res, next){
    const age = req.query.age;
    if(age >= 14){
        next();
    }else{
        res.json({
            msg:"Not eligible"
        })
    }
}

// just pass / define the middleware to the request

app.get("/ride1", age14OrMoreMiddleware, function(req, res){
    res.json({
        msg : "successfully ridden ride1",
    })
})

app.get("/ride2", age14OrMoreMiddleware, function(req, res){
    res.json({
        msg : "successfully ridden ride2",
    })
```

```
})


app.listen(3000);
```

:sparkle: **Explanation of the below code (Important)**

```
app.get("/ride1", age14OrMoreMiddleware, function(req, res){
    res.json({
        msg : "successfully ridden ride1",
    })
})
```

you have **Defined** the series that before `function (req, res){....}` is called , `age14OrMoreMiddleware` **function should be called and hence series of middleware means.. in** `express`

Now to refine the above code

Another way to use the middleware (`.use` method)

> :sparkle: **If you know certain** `middleware` **is going to be used in every** `route` **then instead of passing it on each** `route` **one by one use** `app.use()` **method**

for example -> above `age14OrMoreMiddleware` was being used to both the `routes` so instead of passing them on all `routes` one by one just use `app.use()`

```
const express = require("express")

const app = express();

function age14OrMoreMiddleware(req, res, next){
    const age = req.query.age;
    if(age >= 14){
        next();
    }else{
        res.json({
            msg:"Not eligible"
        })
    }
}

// Just use this line of code to make available the middleware to EVERY method

app.use(age14OrMoreMiddleware)

// The above passed middleware will automatically passed to all the method defined
below
```

```
app.get("/ride1", function(req, res){
    res.json({
        msg : "successfully ridden ride1",
    })
})

app.get("/ride2", function(req, res){
    res.json({
        msg : "successfully ridden ride2",
    })
})


app.listen(3000);
```

:sparkle: **2nd Most important point -> while using `app.use()` is yaad rkhna "Jahan par v `app.use()` likha hoga uske baad ke jitne `middleware` h usi me apply hoga ya pass on hoga"**

```
const express = require("express")

const app = express();

function age14OrMoreMiddleware(req, res, next){
    const age = req.query.age;
    if(age >= 14){
        next();
    }else{
        res.json({
            msg:"Not eligible"
        })
    }
}

app.get("/ride1", function(req, res){
    res.json({
        msg : "successfully ridden ride1",
    })
})

app.use(age14OrMoreMiddleware) // as "/ride1" ke baad use kiya gya and "/ride2" ke upar to bas "/ride2" aur iske neeche jo v hoga sb pe pass on hoga

app.get("/ride2", function(req, res){
    res.json({
        msg : "successfully ridden ride2",
    })
})

app.use(age14OrMoreMiddleware) // If you write at LAST then there is no use of middleware made
```

```
app.listen(3000);
```

Now lets go to the **Assignment section**

you can go directly here -> Middleware Assignment

and then go to the **readme.md** file, where you will get to know about what you have to do

1st Assignment

💡 **How to make a `Global middleware` which count the number of request coming to the server ??**

-> `Global Middleware` -> `middleware` which is applicable to all the `routes` written in the code (example is above function `age14OrMoreMiddleware` is `global middleware`)

```
let requestCount = 0

// you can directly pass the function inside .use() instead of declaring it above
and then referencing it

app.use(function(req, res, next){
    requestCount = requestCount + 1;
    next();
})
```

2nd Assignment

```
// You have been given an express server which has a few endpoints.
// Your task is to create a global middleware (app.use) which will
// rate limit the requests from a user to only 5 request per second
// If a user sends more than 5 requests in a single second, the server
// should block them with a 404.
// User will be sending in their user id in the header as 'user-id'
// You have been given a numberOfRequestsForUser object to start off with which
// clears every one second
```

The above thing is very helpful if suppose someone wants to bombard your website with too many request, it will limit it the request

> :sparkle: **for capturing something in the `headers` use `req.headers[""]`**

```
let numberOfRequestForUser = {}

// see the problem and then first divide the problem
// 1. How to count number of request present in the headers ??
// and 2. How to count requests IN A SINGLE SECOND and clears after 1 second ??

// solving the 1. problem
app.use(function(req, res, next){
    numberOfRequestForUser[req.headers["user-id"]]++;
```

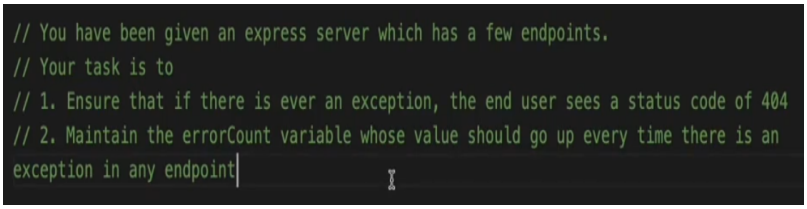
```
})

// solving the 2. problem
// after 1 second it will again point to EMPTY OBJECT

setInterval(() => {
    numberOfRequestForUser = {};
}, 1000);

// making the above code more cleaner and aligned to the question asked
app.use(function(req, res, next){
    const userID = req.headers["user-id"]; // checking who is the user coming ??
    if(numberOfRequestForUser[userID]){
        numberOfRequestForUser[userID] = numberOfRequestForUser[userID] + 1;
        // now checking number of request and then limiting the rate
        if(numberOfRequestForUser > 5){
            res.status(404).send("No Entry further");
        }else{ // do the further things
            next();
        }
    }else{ // agar pehli baar aa rha h userID to usko to register krna he pdega
and then initialise it with 1
        numberOfRequestForUser[userID] = 1;
        next();
    }
})
```

3rd Assignment



About `throw new ERROR()`

basically when you do something which is **not legal** according to the `javascript` langauge, **it throws error** for example ->

```
let a;

a.length(); // running this line js will THROW AN ERROR
```

To throw this error, `js` gives you a way which is `throw new ERROR` command

```
const express = require("express")
```

```
const app = express();

app.get('/user', function(req, res){
    throw new Error("User not found");
    res.status(200).json({name : "John"});
})

app.post('/user', function(req, res){
    res.status(200).json({msg : "created dummy user"});
})

app.get('/errorCount', function(req, res){
    res.status(200).json({errorCount});
})

app.listen(3000);
```

Problem with the above code

```
Error: User not found
    at /Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/03-errorcount.js:14:9
    at Layer.handle [as handle_request] (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/layer.js:95:5)
    at next (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/route.js:144:13)
    at Route.dispatch (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/route.js:114:3)
    at Layer.handle [as handle_request] (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/layer.js:95:5)
    at /Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/index.js:284:15
    at Function.process_params (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/index.js:346:12)
    at next (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/index.js:280:10)
    at expressInit (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/middleware/init.js:40:5)
    at Layer.handle [as handle_request] (/Users/kirat/Projects/100xdevs/cohort2/assignments/week-3/01-middlewares/node_modules/express/lib/router/layer.js:95:5)
```

although it throws an error in exception it **revealed a lot of info. to the user which is not good** also if you `inspect` this code in **browser** you will see that it is throwing status code `500`

> :sparkle: **By default ->** `js` when throws an expception it throws it on **Status code** `500`

To fix it `express` gives a special type of `middleware`

Error-handling Middleware

What is error-handling middleware ??

-> **Anytime there is an exception being thrown** for example ->

```
let a;  // although "a" is undefined
let b = a.length; // you are trying to access "a" this is EXCEPTION
```

comes role of `error-handling middleware`

> ⚠️ **Always remember to write `error-handling middleware` at the end of the code**

How to define `error-handling middleware` ??

It looks very similar to normal `middleware` just that it has `4 Arguments`

🔷 **The biggest advantage of using this is you get to use Status Code of your choice**

Which is not in the case of default `js` exception (`status code -> 500`)

```
app.use(function(err, req, res, next){
    res.status(404).send({});
})
```

now finally doing the assignment

```
const express = require("express")

const app = express();

let errorCount = 0;

app.get('/user', function(req, res){
    throw new Error("User not found");
    res.status(200).json({name : "John"});
})

app.post('/user', function(req, res){
    res.status(200).json({msg : "created dummy user"});
})

app.get('/errorCount', function(req, res){
    res.status(200).json({errorCount});
})

// Error - Handling Middleware

app.use(function(err, req, res, next){
    res.status(404).send({});
    errorCount = errorCount + 1; // as the assignment says whenever you hit an
error you should increase it
})

app.listen(3000);
```